

IMPORTANT

Please note that this guide is still largely a work in progress.

This is written by work study student Christopher Deng, so [contact me](#) if you have any suggestions or questions about this guide.

~~Unfortunately, since this is a Markdown file, there is no easier way to 'collaborate' on this. With that being said, you can always email me with additions or fixes.~~

UPDATE: this Markdown file is now available publicly on [GitHub](#), feel free to suggest changes

I will keep uploading the updated .pdf files to Microsoft Teams > New College IT > General > Files

Introduction

This documentation will walk you through the installation of Ubuntu 24.04 (Noble Numbat) server (without a GUI) in a VirtualBox OR UTM virtual machine.

Additionally, we'll cover how to ensure network connectivity and how to fully deploy a WordPress generic site on LAMP (Linux, Apache, MySQL, Perl/PHP/Python).

Installing VirtualBox or UTM


The downloads can all be found on the [VirtualBox Downloads](#) page. Ensure that you download the latest version.

Windows

1. Visit the [VirtualBox Downloads](#) page
2. Look for 'Windows hosts' and click on the link to download

MacOS (READ FIRST)

For Mac systems, please first check whether you're using Apple Silicon or Intel:

1. On the top left of your screen, click the  Apple Menu and select 'About This Mac'
2. Look at 'Chip' or 'Processor'. If it says something like M1, M2, or M3, you're on Apple Silicon. Otherwise, 'Intel' indicates that it's running an Intel processor.

MacOS (Apple Silicon)

VirtualBox does not work well with Apple Silicon. Instead, we can use UTM.

1. Visit the [UTM Downloads](#) page
2. Choose whichever method you prefer; I did the direct download instead of app store.

MacOS (Intel)

1. Visit the [VirtualBox Downloads](#) page
2. Look for macOS / Intel hosts' and click on the link to download

Creating and Configuring the VirtualBox Virtual Machine

Download Ubuntu 24.04 LTS Server install image

1. Go to the [Ubuntu 20.04 downloads](#) page and click '64-bit PC (AMD64) server install image' to download the ISO.

Create a new Virtual Machine

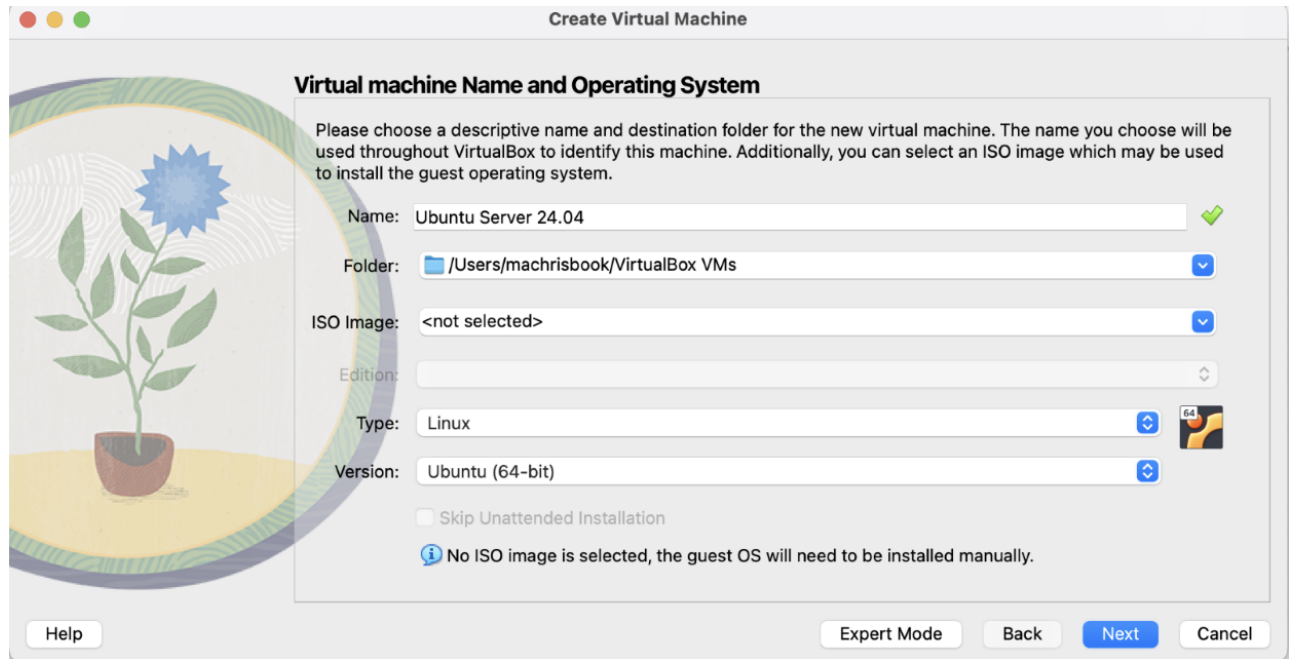
1. Open VirtualBox and click 'New'



2. Virtual machine Name and Operating System

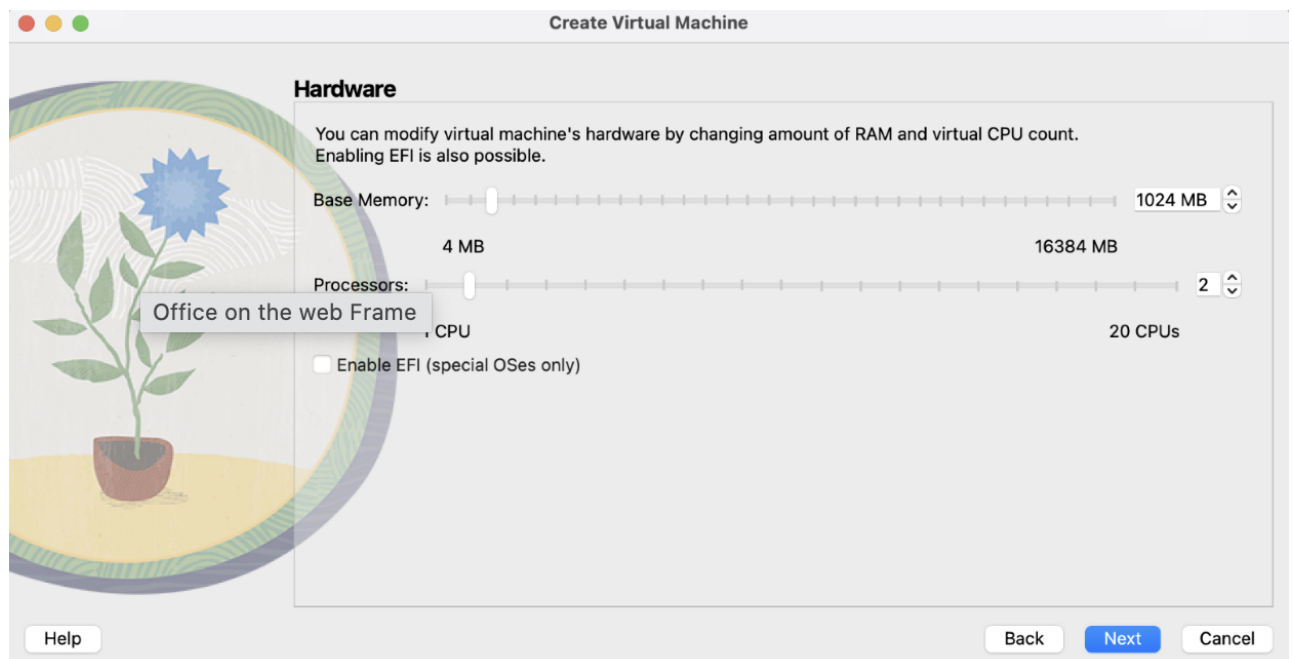
Fill in the fields appropriately. First, give the Virtual Machine a name, such as 'Ubuntu Server 24.04'. This should automatically fill out the rest of the fields, such as Type and Version. Otherwise, match

the fields appropriately. You can leave the default folder, and do not mount any ISOs just yet.



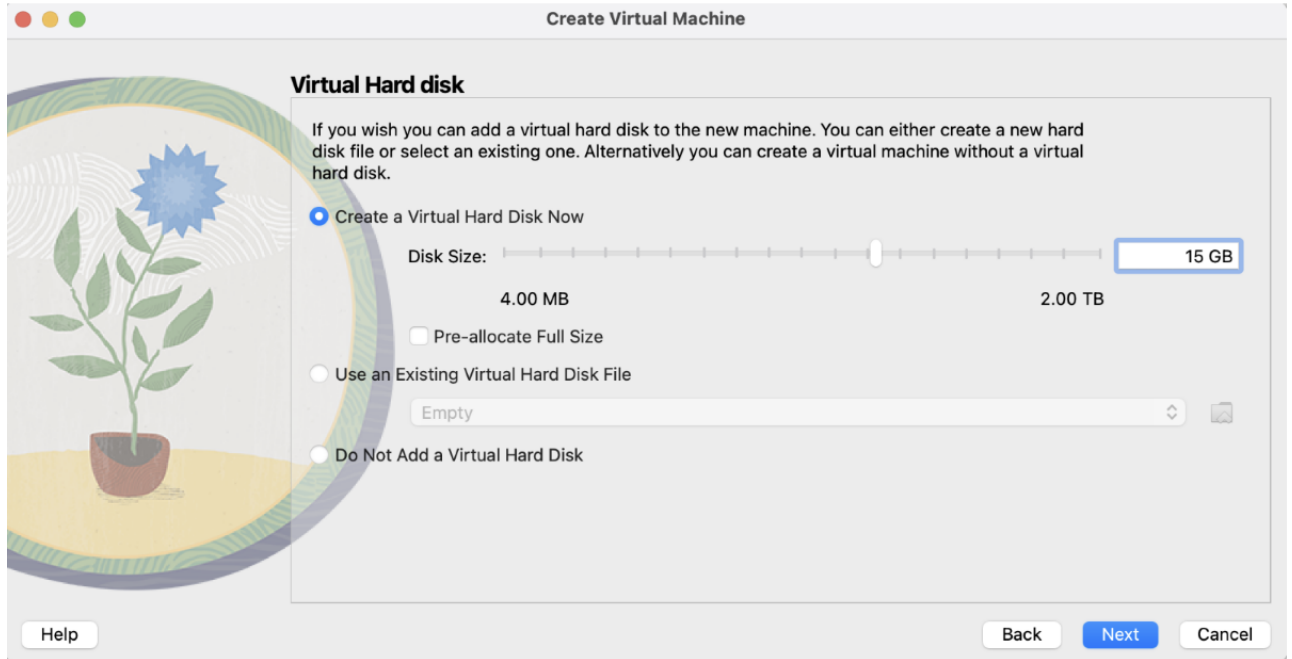
3. Hardware

- Base Memory:** 1GB is enough
- Processors:** 1 processor should also suffice, but you can add more if you want (for better performance)
- Enable EFI:** for simplicity (especially for a GUI-less server), we'll leave it off



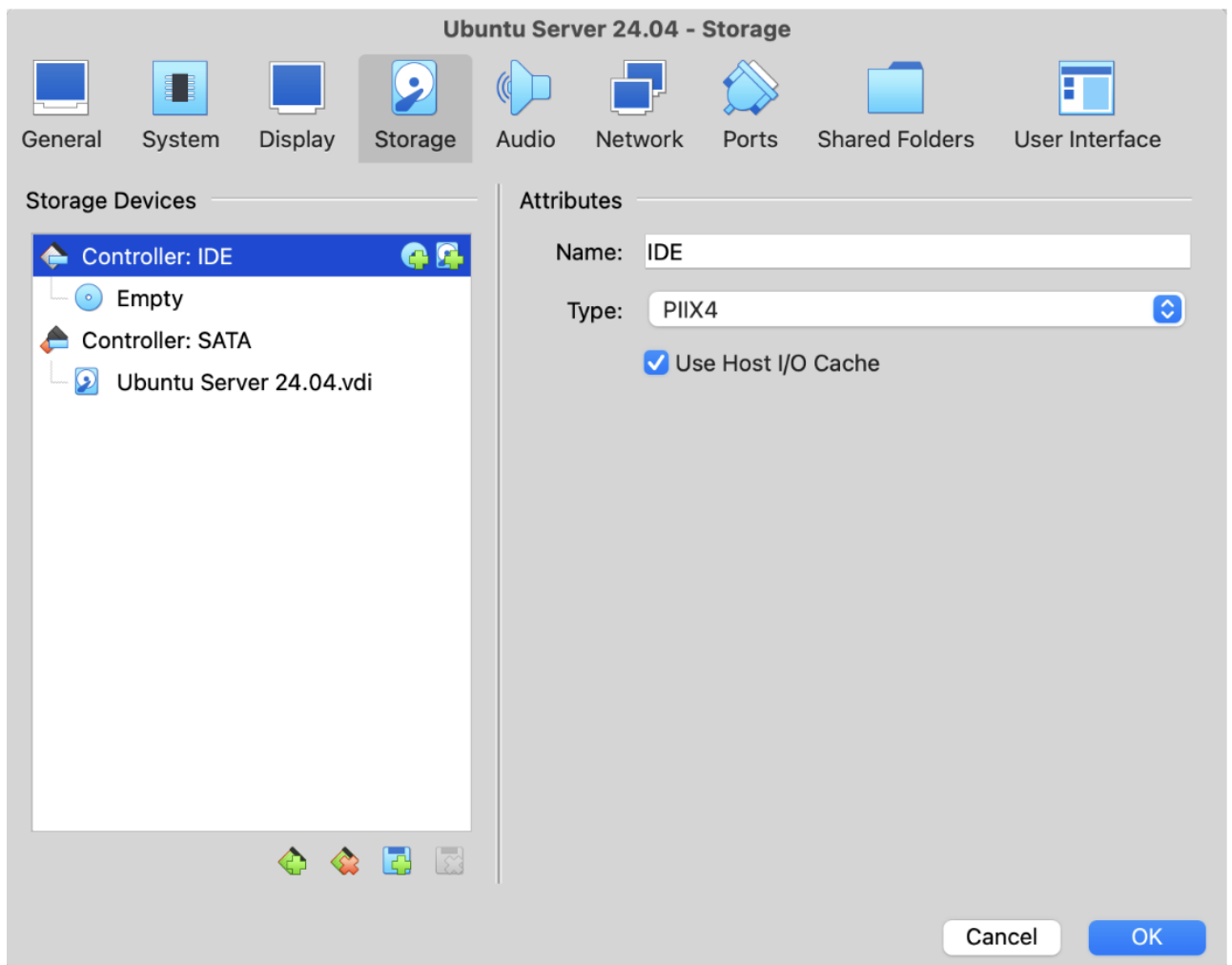
4. Virtual Hard Disk

You want to Create a Virtual Hard Disk Now. Set it to at least 10 GB.

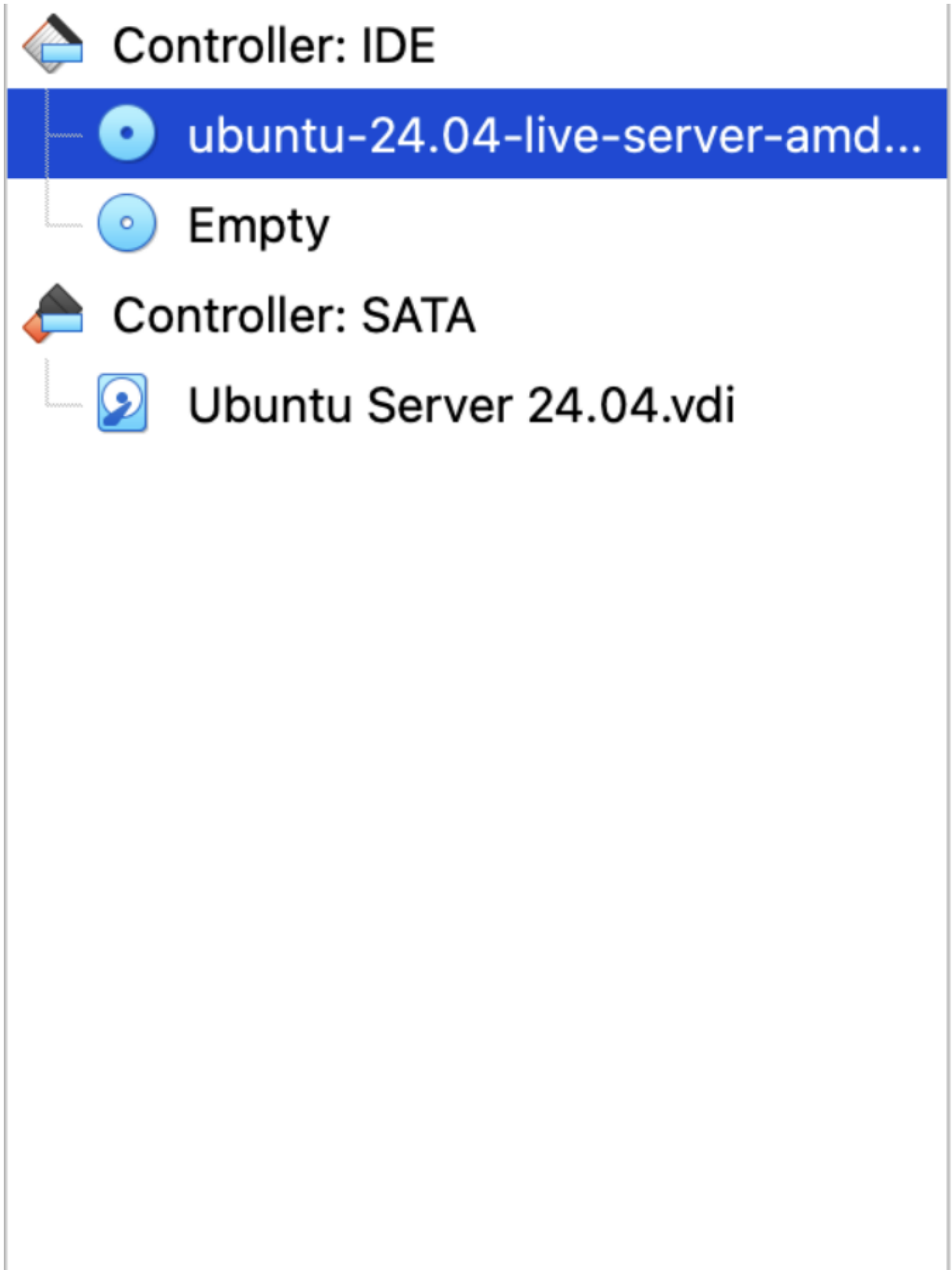


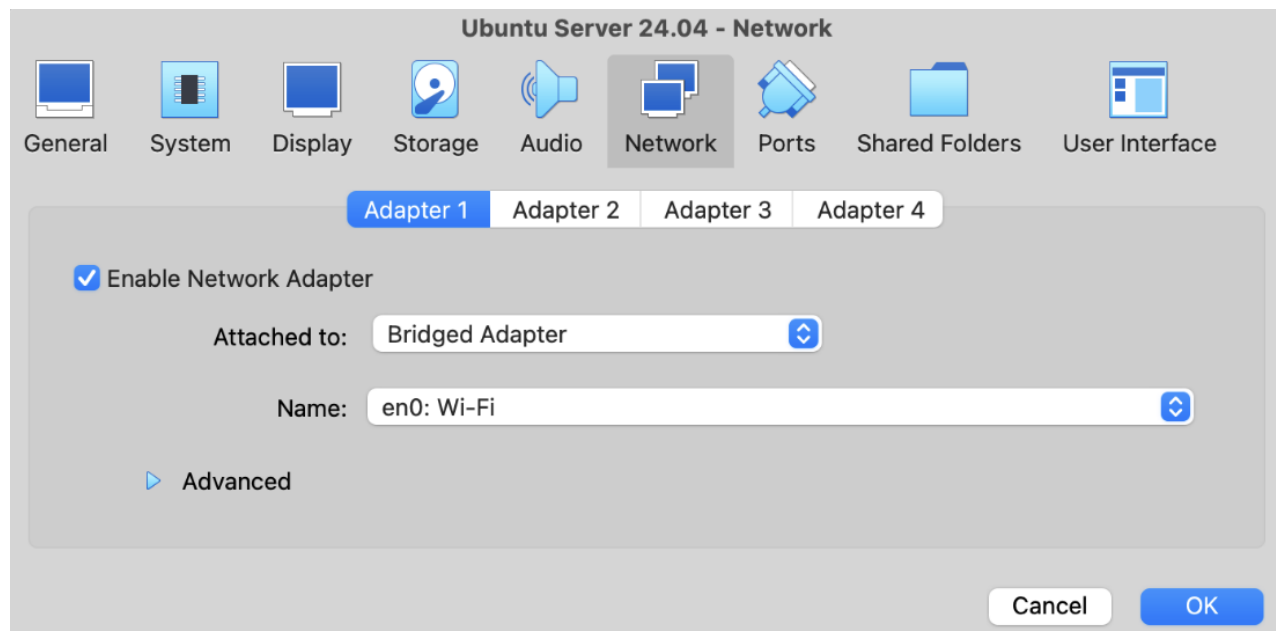
5. After you finish the virtual machine creation wizard, you should be back at the home page. With your newly created VM, click on the 'Settings' cog to change the Storage and Network fields.

a. Storage



Click on the empty optical drive and attach the Ubuntu ISO you downloaded earlier. Your list should now look like this:

**b. Network**



Change the Attached to field to 'Bridged Adapter' for network access.

For the Name, choose whatever you're connected to:

en0 – choose this if you're connected via Wi-Fi

en1, en2, en3 – choose this if you're using a Thunderbolt Ethernet adapter

en4, en5, en6 – choose this if you're connected via Ethernet

Installing Ubuntu 20.04 LTS Server

Work in progress...

Verifying Network Connectivity

Run the following command to send an infinite amount of packets, and press CTRL + C to stop sending packets.

```
ping.google.com
```

```
notexploiting@ubuntu-server-utm:~$ ping google.com
PING google.com (142.250.191.142) 56(84) bytes of data:
64 bytes from ord38s29-in-f14.1e100.net (142.250.191.142): icmp_seq=1 ttl=117 time=16.4 ms
64 bytes from ord38s29-in-f14.1e100.net (142.250.191.142): icmp_seq=2 ttl=117 time=16.2 ms
64 bytes from ord38s29-in-f14.1e100.net (142.250.191.142): icmp_seq=3 ttl=117 time=20.1 ms
64 bytes from ord38s29-in-f14.1e100.net (142.250.191.142): icmp_seq=4 ttl=117 time=17.6 ms
64 bytes from ord38s29-in-f14.1e100.net (142.250.191.142): icmp_seq=5 ttl=117 time=17.1 ms
64 bytes from ord38s29-in-f14.1e100.net (142.250.191.142): icmp_seq=6 ttl=117 time=18.8 ms
64 bytes from ord38s29-in-f14.1e100.net (142.250.191.142): icmp_seq=7 ttl=117 time=16.3 ms
^C
--- google.com ping statistics ---
7 packets transmitted, 7 received, 0% packet loss, time 6019ms
rtt min/avg/max/mdev = 16.201/17.501/20.092/1.368 ms
```

Alternatively, you can run the following command with the `-c` flag to send a specified number of packets.

```
ping -c 3 google.com
```

```
notexploiting@ubuntu-server-utm:~$ ping -c 3 google.com
PING google.com (142.250.191.142) 56(84) bytes of data.
64 bytes from ord38s29-in-f14.1e100.net (142.250.191.142): icmp_seq=1 ttl=117 time=18.8 ms
64 bytes from ord38s29-in-f14.1e100.net (142.250.191.142): icmp_seq=2 ttl=117 time=18.3 ms
64 bytes from ord38s29-in-f14.1e100.net (142.250.191.142): icmp_seq=3 ttl=117 time=16.2 ms

--- google.com ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2006ms
rtt min/avg/max/mdev = 16.243/17.783/18.820/1.110 ms
```

Next, we can also send packets to 8.8.8.8 in the same manner.

```
ping 8.8.8.8
```

```
notexploiting@ubuntu-server-utm:~$ ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=112 time=19.3 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=112 time=19.5 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=112 time=22.5 ms
64 bytes from 8.8.8.8: icmp_seq=4 ttl=112 time=19.9 ms
64 bytes from 8.8.8.8: icmp_seq=5 ttl=112 time=23.7 ms
64 bytes from 8.8.8.8: icmp_seq=6 ttl=112 time=22.4 ms
64 bytes from 8.8.8.8: icmp_seq=7 ttl=112 time=24.0 ms
^C
--- 8.8.8.8 ping statistics ---
7 packets transmitted, 7 received, 0% packet loss, time 6022ms
rtt min/avg/max/mdev = 19.283/21.599/24.004/1.858 ms
```

```
ping -c 3 8.8.8.8
```

```
notexploiting@ubuntu-server-utm:~$ ping -c 3 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=112 time=22.9 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=112 time=23.9 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=112 time=19.1 ms

--- 8.8.8.8 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2005ms
rtt min/avg/max/mdev = 19.080/21.939/23.876/2.063 ms
```

Deploy WordPress on LAMP Stack

Installing Apache and Updating the Firewall

Start by updating the package manager cache.

If it asks for the password, enter your password (side note: it may not look like it's entering, but it is)

```
sudo apt update
```

Then, install Apache with:

```
sudo apt install apache2
```

When prompted to confirm Apache's installation, confirm by typing **Y**, then **ENTER**.

Once the installation is finished, you'll need to adjust your firewall settings to allow HTTP traffic. We'll be leveraging the different application profiles in Ubuntu's default firewall configuration tool, Uncomplicated Firewall (UFW).

List all currently available UFW application profiles:

```
sudo ufw app list
```

Here's what each of the profiles mean:

- **Apache**: This profile opens only port **80** (normal, unencrypted web traffic).
- **Apache Full**: This profile opens both port 80 (normal, unencrypted web traffic) and port 443 (TLS/SSL encrypted traffic).
- **Apache Secure**: This profile opens only port 443 (TLS/SSL encrypted traffic).

For now, it's best to allow only connections on port 80, since this is a fresh Apache installation and you don't yet have a TLS/SSL certificate configured to allow for HTTPS traffic on your server.

Only allow traffic on port **80**, use the **Apache** profile:

```
sudo ufw allow in "Apache"
```

Verify the change with:

```
sudo ufw status
```


If your output says **Status: inactive**, run the following command to enable **ufw**:

```
sudo ufw enable
```

Run **sudo ufw status** once again and hopefully you see something like the following:

```
notexploiting@ubuntu-server-utm:~$ sudo ufw status
Status: active

To Action From
--
Apache ALLOW Anywhere
Apache (v6) ALLOW Anywhere (v6)
```

Ensure that Apache is running and active on the server:

```
sudo systemctl status apache2
```

```
notexploiting@ubuntu-server-utm:~$ sudo systemctl status apache2
• apache2.service - The Apache HTTP Server
  Loaded: loaded (/usr/lib/systemd/system/apache2.service; enabled; preset: enabled)
  Active: active (running) since Thu 2024-06-20 18:58:12 UTC; 23min ago
    Docs: https://httpd.apache.org/docs/2.4/
  Main PID: 9094 (apache2)
    Tasks: 55 (limit: 2272)
  Memory: 5.4M (peak: 5.7M)
    CPU: 1.159s
  CGroup: /system.slice/apache2.service
          └─9094 /usr/sbin/apache2 -k start
             └─9097 /usr/sbin/apache2 -k start
                └─9098 /usr/sbin/apache2 -k start
```

If Apache is not running, start it:

```
sudo systemctl start apache2
```

Now we can do a spot check. First, get your public IP address:

```
curl ifconfig.me
```

There should be a string of numbers before your server username. With this IP address in mind, go to your browser and enter the following address:

http://your_server_ip

You should be able to see the default Ubuntu Apache web page. For example:



Apache2 Default Page

Ubuntu

It works!

This is the default welcome page used to test the correct operation of the Apache2 server after installation on Ubuntu systems. It is based on the equivalent page on Debian, from which the Ubuntu Apache packaging is derived. If you can read this page, it means that the Apache HTTP server installed at this site is working properly. You should **replace this file** (located at `/var/www/html/index.html`) before continuing to operate your HTTP server.

If you are a normal user of this web site and don't know what this page is about, this probably means that the site is currently unavailable due to maintenance. If the problem persists, please contact the site's administrator.

Configuration Overview

Ubuntu's Apache2 default configuration is different from the upstream default configuration, and split into several files optimized for interaction with Ubuntu tools. The configuration system is **fully documented in** `/usr/share/doc/apache2/README.Debian.gz`. Refer to this for the full documentation. Documentation for the web server itself can be found by accessing the **manual** if the `apache2-doc` package was installed on this server.

The configuration layout for an Apache2 web server installation on Ubuntu systems is as follows:

```
/etc/apache2/
|-- apache2.conf
|   |-- ports.conf
|-- mods-enabled
|   |-- *.load
|   |-- *.conf
|-- conf-enabled
|   |-- *.conf
|-- sites-enabled
|   |-- *.conf
```

- `apache2.conf` is the main configuration file. It puts the pieces together by including all remaining configuration files when starting up the web server.
- `ports.conf` is always included from the main configuration file. It is used to determine the listening ports for incoming connections, and this file can be customized anytime.
- Configuration files in the `mods-enabled/`, `conf-enabled/` and `sites-enabled/` directories contain particular configuration snippets which manage modules, global configuration fragments, or virtual host configurations, respectively.
- They are activated by symlinking available configuration files from their respective `*-available/` counterparts. These should be managed by using our helpers `a2enmod`, `a2dismod`, `a2ensite`, `a2dissite`, and `a2enconf`, `a2disconf`. See their respective man pages for detailed information.
- The binary is called `apache2` and is managed using `systemd`, so to start/stop the service use `systemctl start apache2` and `systemctl stop apache2`, and use `systemctl status apache2` and `journalctl -u apache2` to check status. `system` and `apache2ctl` can also be used for service management if desired. **Calling `/usr/bin/apache2` directly will not work** with the default configuration.

Document Roots

By default, Ubuntu does not allow access through the web browser to *any* file outside of those located in `/var/www`, `public_html` directories (when enabled) and `/usr/share` (for web applications). If your site is using a web document root located elsewhere (such as in `/srv`) you may need to whitelist your document root directory in

If you can view this page, your web server is correctly installed and accessible through your firewall.

However, you might not be able to load it completely (which is the case for me, since I'm using student residence wifi). You can instead try to load the page directly in the command line:

```
curl http://localhost
```

You should be able to see a bunch of HTML lines, like this:

```
                default configuration.
            </li>
        </ul>
    </div>

    <div class="section_header">
        <div id="docroot"></div>
        Document Roots
    </div>

    <div class="content_section_text">
        <p>
            By default, Ubuntu does not allow access through the web browser to
            <em>any</em> file outside of those located in <tt>/var/www</tt>,
            <a href="http://httpd.apache.org/docs/2.4/mod/mod_userdir.html" rel="nofollow">public_html</a>
            directories (when enabled) and <tt>/usr/share</tt> (for web
            applications). If your site is using a web document root
            located elsewhere (such as in <tt>/srv</tt>) you may need to whitelist your
            document root directory in <tt>/etc/apache2/apache2.conf</tt>.
        </p>
        <p>
            The default Ubuntu document root is <tt>/var/www/html</tt>. You
            can make your own virtual hosts under /var/www.
        </p>
    </div>

    <div class="section_header">
        <div id="bugs"></div>
        Reporting Problems
    </div>
    <div class="content_section_text">
        <p>
            Please use the <tt>ubuntu-bug</tt> tool to report bugs in the
            Apache2 package with Ubuntu. However, check <a
            href="https://bugs.launchpad.net/ubuntu/+source/apache2"
            rel="nofollow">existing bug reports</a> before reporting a new bug.
        </p>
        <p>
            Please report bugs specific to modules (such as PHP and others)
            to their respective packages, not to the web server itself.
        </p>
    </div>
</div>
<div class="validator">
</div>
</body>
</html>
notexploiting@ubuntu-server-utm:~$ _
```

Installing MySQL

You'll need to install a database system to be able to store and manage data for your site. MySQL is a popular database management system used within PHP environments

Install this software with **apt**:

```
sudo apt install mysql-server
```

When prompted to confirm the installation, confirm by typing **Y**, then **ENTER**.

Now we want to run a security script that removes some insecure default settings and lock down access to your database system.

`mysql_secure_installation` runs in a recursive loop, so we'll need to first adjust how your root MySQL user authenticates.

First, open up the MySQL Prompt:

```
sudo mysql
```

You should now see that the current line only says `mysql>`

Now change the **root** user's authentication to one that uses a password

```
ALTER USER 'root'@'localhost' IDENTIFIED WITH mysql_native_password BY  
'password';
```

After making this change, exit the MySQL prompt:

```
exit
```

Now run the script (hopefully now without issue):

```
mysql_secure_installation`
```

When prompted, enter your password. By default (from the script above), the password is 'password'.

Read more on this [DigitalOcean tutorial](#) on the information regarding passwords and **VALIDATE PASSWORD COMPONENT**.

For the rest of the questions, press 'Y' and hit the 'ENTER' key at each prompt. This will remove some anonymous users and the test database, disable remote root logins, and load these new rules so that MySQL immediately respects the changes you have made. (I didn't do this because I didn't read the tutorial)

Securing the MySQL server deployment.

Enter password for user root:

VALIDATE PASSWORD COMPONENT can be used to test passwords and improve security. It checks the strength of password and allows the users to set only those passwords which are secure enough. Would you like to setup VALIDATE PASSWORD component?

Press y|Y for Yes, any other key for No: no

Using existing password for root.

Change the password for root ? ((Press y|Y for Yes, any other key for No) : no

... skipping.

By default, a MySQL installation has an anonymous user, allowing anyone to log into MySQL without having to have a user account created for them. This is intended only for testing, and to make the installation go a bit smoother. You should remove them before moving into a production environment.

Remove anonymous users? (Press y|Y for Yes, any other key for No) : no

... skipping.

Normally, root should only be allowed to connect from 'localhost'. This ensures that someone cannot guess at the root password from the network.

Disallow root login remotely? (Press y|Y for Yes, any other key for No) : yes
Success.

By default, MySQL comes with a database named 'test' that anyone can access. This is also intended only for testing, and should be removed before moving into a production environment.

Remove test database and access to it? (Press y|Y for Yes, any other key for No) : no

... skipping.

Reloading the privilege tables will ensure that all changes made so far will take effect immediately.

Reload privilege tables now? (Press y|Y for Yes, any other key for No) : yes
Success.

All done!